FOR ATARI® COMPUTERS

# INSTEDIT


**WE MAKE USING COMPUTERS FUN!**

ESI
PROGRAM EXCHANGE
EDUCATIONAL SOFTWARE inc.

**INSTEDIT**

by Sheldon Leemon


**How to Load**

**TAPE....**

Place the tape in your recorder, label side up. Make sure the tape is rewound, and reset the counter to zero. Push PLAY on the recorder, type CLOAD and press RETURN twice. If the program won't start to load, try positioning the tape forward or backwards a little. A little trick to find the beginning of the program is to first, turn your volume up. Then POKE 54018,52 to start the cassette motor. Listen to the "noise" on the tape. When you find the high pitched steady tone, you have found the beginning of the program. POKE 54018,60 to turn the cassette motor off. You can now LOAD the program as shown above. When the READY prompt appears, type RUN and press RETURN.

**DISK....**

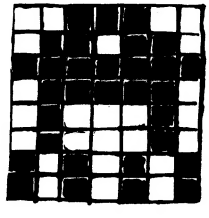
To LOAD and RUN the disk, first turn on your disk drive. When the busy light goes out, place the disk in the drive. Now turn on the computer, with the BASIC Cartridge in place the program will LOAD each part and RUN by itself.

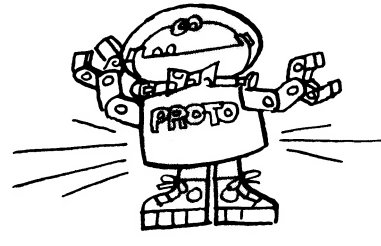
**Requirements:**

ATARI BASIC Cartridge
ATARI 810 Disk Drive (with 24K of memory) or
ATARI 410 Program Recorder(with 16K)
One Joystick controller

## CONTENTS



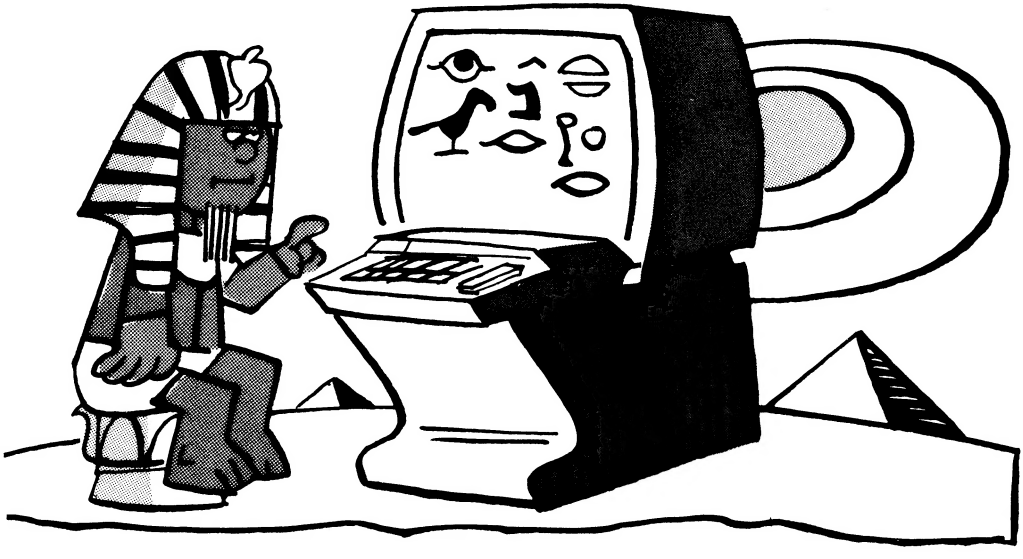




## INTRODUCTION

Why create custom character sets?

You could have a thousand reasons for adding custom characters to your programs; you may need special math symbols, chemistry symbols, foreign language alphabets, or a special typeface to set the mood for a particular program.



On a more sophisticated level, you may want to develop special graphics characters as a substitute for the plot-and-draw method for producing high-resolution graphics. By positioning several characters together, you can create a high-res picture as easily as you can print a character string, like this:

```
PRINT "ABC123"
```

Using machine-language subroutines, you can extend your work into developing professional-quality, fast-action arcade games with full color, high-resolution animation. Using custom character sets in this way introduces a new level of graphics power and flexibility.

You can achieve the same resolution with custom mode 0 characters as you can with mode 8 drawing. The difference is that by using specially designed characters, you avoid the high memory usage required for mode 8, as well as the tedious and slow plotting of each point and line. You can put the finished drawing on the screen much more quickly, achieving smooth animation in BASIC just by printing strings of characters in succession, each representing a figure in a different position.

If you're familiar with the ATARI computer's player missile graphics system, you can even use INSTEDIT to design players.

**HOW INSTEDIT WORKS**

PRESSING A KEY

When you press a key on your ATARI computer's keyboard, one of 128 different characters appears on your TV screen. The shape of each character is determined by a collection of data stored in computer memory. INSTEDIT lets you change this data, and thus the shape of the characters. as easily as drawing a picture. For a more thorough description of this process, see our TRICKY TUTORIAL #8 - Character Graphics.

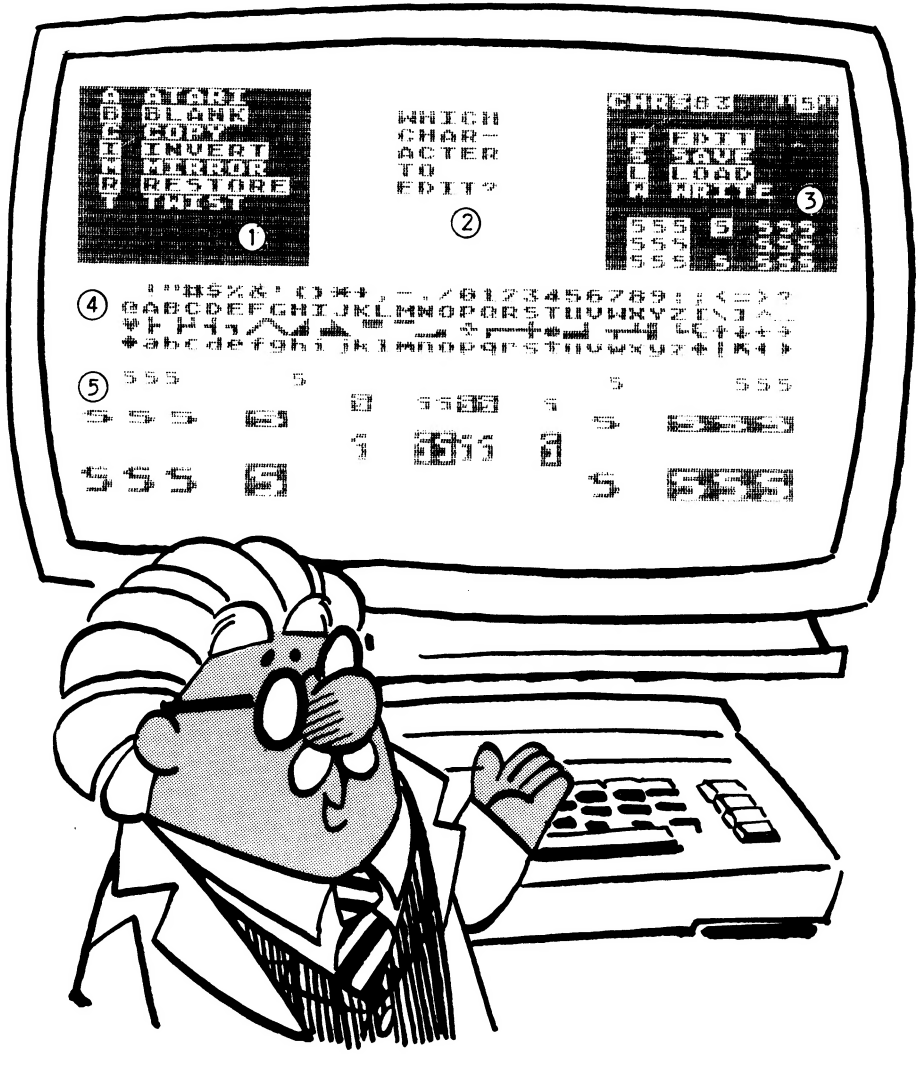
Here's what to do. Use the computer's keyboard to select the character you want to edit and a large-scale model of this character is displayed in an eight-by-eight grid.

Using a joystick, move the cursor to any of the 64 squares and modify the character's shape--add or remove color by pressing the red trigger button (if color is already present, pressing the button erases it; if color is absent, pressing the button adds color). As you change the model character, these changes appear in the samples of the character, which are displayed in each character mode. Additional editing features let you experiment further with your characters. You can save your finished set on diskette or cassette for future editing, or you can incorporate it into a BASIC or assembly language program.

Let's take a look at the menu of INSTANT EDIT so you can see the power you have over the ATARI character set.

## INSTEDIT'S MENU

INSTEDIT's display screen is divided into five areas. Here's what it looks like:



(1) Submenu of advanced editing features, usable with the main menu's EDIT option.

(2) Frame for the 8-x-8 character grid. Initially, this frame contains the program title. Prompts also display in this area.

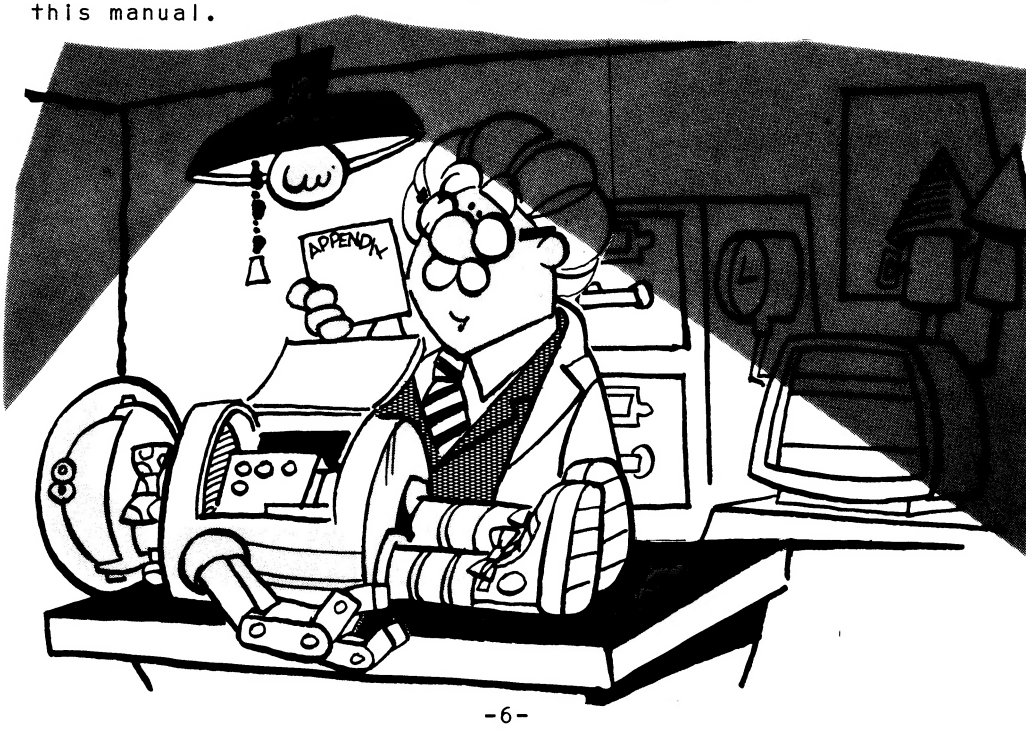
(3) Main menu of program options. Underneath the options are three sample rows of the model character shown using the question mark (?) in mode 0, in both normal and inverse video.

(4) Mode 0 display of the entire character set (in Internal Character Set order).

(5) The model character (?) displayed in the other five character modes, in this order:

(1) Instruction Register (IR) Mode 3
(2) IR Mode 4
(3) IR Mode 6 (BASIC GR. Mode 1)
(4) IR Mode 5
(5) IR Mode 7 (BASIC GR. Mode 2)

For further information, see the appendix at the end of this manual.



## MENU SELECTIONS

You select a menu function by pressing the key matching the first letter of the function. Here are your choices:

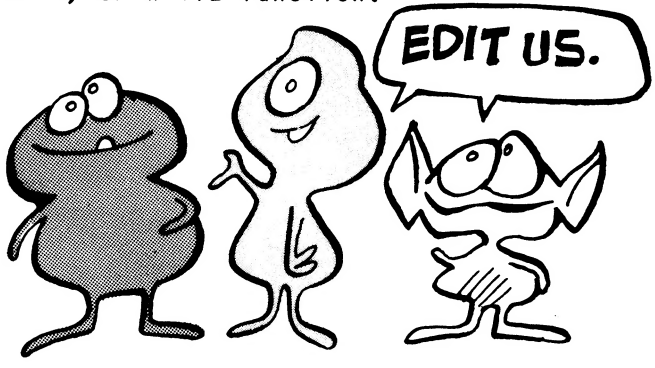
### E EDIT

When you press the letter "E", the prompt, WHICH CHARACTER TO EDIT?, displays in the center frame. Type in your desired character, using the LOWR or CTRL key, as appropriate, to edit a lower case or control character. If your character is an editing symbol, such as one of the arrows, you don't need to press the ESC key before pressing the combination of keys used to carry out the edit function.

After you select a character to edit, your choice is displayed at the top of the main menu and in all of the screen locations initially occupied by the question mark. A large-scale model is also displayed in the center frame, along with a yellow cursor.

Use your joystick to move the cursor within the frame to edit the character. Press the red trigger button to change the square. If the square is empty, pressing the buton will fill it; if the square is filled, pressing the red trigger will empty it. Each of the sample characters reflects your changes as soon as they occur; however, the menu and prompt lettering remain in the standard character set.

While using the EDIT mode, you can also use the submenu options of ATARI, BLANK, COPY, INVERT, MIRROR, RESTORE, and TWIST. After you finish editing a character, you can go to another by pressing the "E" again, or you can select the SAVE, LOAD, or WRITE function.



Here are the Submenu functions. Select a function by pressing the matching letter. Remember, you must be in the EDIT mode to use these commands.

A ATARI

Restores the character being edited to its normal (ROM set) appearance.

B BLANK

Clears the whole character currently being edited, giving you a clean background on which to design a new character.

C COPY

Replaces the character currently being edited with any other character. After pressing "C", the prompt, WHICH CHARACTER TO COPY?, displays in the center frame. Type in your desired character. The current character then becomes a duplicate of that character.

I INVERT

Reverses the colors of the current pattern. All colored squares change to background and vice versa.

M MIRROR

Changes the character to the mirror image of its present shape. If the character is symmetrical, no change is apparent.

R RESTORE

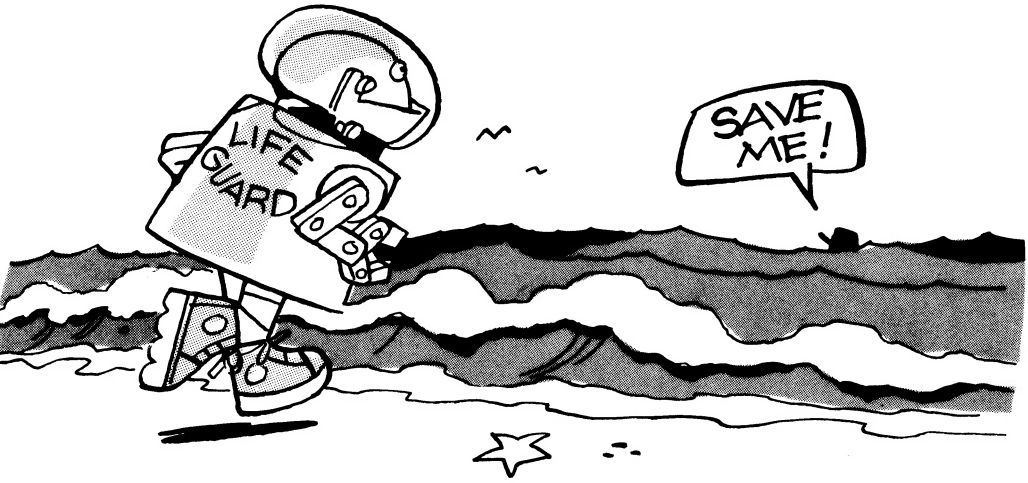
Restores the original character as it appeared the last time you pressed "E" for EDIT. Any intermediate changes are lost.

T TWIST

Rotates the entire character on its axis ninety degrees clockwise each time you press "T".

SHIFT

Depending on which arrow key you press, shifts the whole character up a line, down a line, one square to the left, or one square to the right. You can repeat the function by holding down an arrow key.




## S SAVE

This function is used to store the complete current character set as a data file on diskette or cassette. You can continue editing the set later, or you can incorporate it into another program.

When you press "S", the prompt either asks you to name the data file or to position the cassette, depending on whether a disk drive is connected. Diskette users enter a file name and press the RETURN key. Don't include an extender; the program auomatically adds the extender ".SET" to your file name, to identify it as a character set data file. Cassette users position the tape at a blank spot, press the PLAY and RECORD buttons on their program recorder, and press the RETURN key on the computer keyboard.

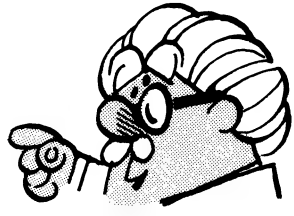
When the program finishes storing the character set to diskette or cassette, the message SAVE COMPLETE displays in the center frame.

If an error occurs so that the program can't complete saving your character set, the original prompt redisplays in the center frame. Try another diskette; the one you're using might contain no free sectors, be damaged, or contain a write-protected file of the same name as the one you're trying to save.

**TO CANCEL A SAVE**

To cancel a SAVE, diskette users should press the RETURN key instead of entering a file name, and cassette users should press the ESC key instead of the RETURN key. In each case, the message CANCEL SAVE displays in the center frame.

Although editing character sets might look like drawing, what is really occurring is plain old data manipulation. As is true any time you work with data, saving your revisions every so often is good practice. This protects you from calamities such as your cat pulling out the cord from the wall outlet, wiping out an afternoon's worth of work. How often you SAVE your work depends on how much you're willing to lose!



**L LOAD**

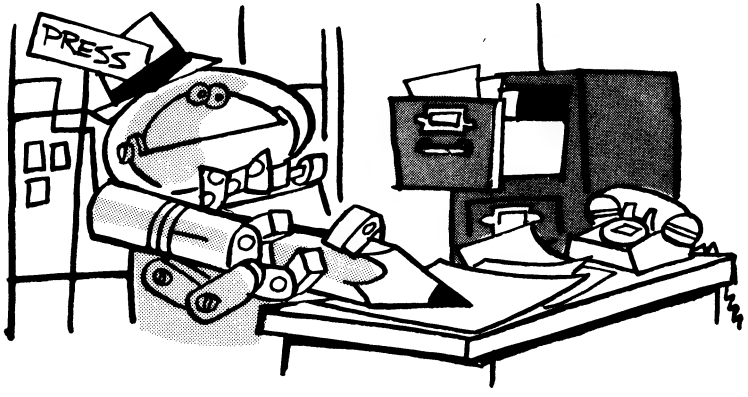
This function will load a previously saved character set from diskette or cassette. If you have a disk drive connected, the program asks you to enter the name of the file to be loaded and then press the RETURN key. Enter only the name you saved; the program automatically adds the extender ".SET" to your file name.

To recall the name of a file, enter an asterisk (*)--the first ten data files in the directory will display at the top of the Submenu area. Display additional file names by pressing the asterisk again.

Cassette users are asked to position the correct tape, press the PLAY button on the program recorder, and press the RETURN key on the computer keyboard.
The message LOAD COMPLETE displays after a successful load.

The ORIGINAL prompt redisplays after an UNSUCCESSFUL load. In the latter case, check that your diskette or cassette containing the desired saved sets are correctly positioned.Diskette users should avoid using any of the I/O commands (Save, Load, or Write) without first making sure their diskette is inserted and the disk drive door is closed. Otherwise, the drive will keep trying to carry out the program's instructions until a timeout occurs. This activity could continue for a couple of minutes, during which time you might think the program has crashed. If one of these situations occurs, just be patient until the timeout occurs and then correct the problem and continue.

Diskette users can cancel this function by pressing the RETURN key without first entering a file name. Cassette users can do so by pressing the ESC key instead of the RETURN key.




## W WRITE

Use this function to store a file on diskette or cassette in such a form that you can incorporate it into an existing program. To provide flexible alternatives for fullest use of your newly defined characters, INSTEDIT can write your character set data in any of three formats. This function is one of INSTEDIT's more powerful features, and you should read this section carefully to understand how to use the WRITE function.

When you press "W", a submenu displays the three WRITE options: (1) BASIC, (2) DATA, and (3) .BYTE. Here's what these options mean:

(1) BASIC

Use this option to write an entire BASIC subroutine to cassette or diskette that integrates your new character set into an existing program. In the INSTEDIT program, the character data is stored as a string. When this string is defined by the program statements of the subroutine, loading the character set from an external storage device isn't necessary. The program loads and runs with your new set. This method uses a relatively large amount of memory, but it's the only way known by the author for instantly installing a whole new character set in a cassette-based program.

The subroutine this option writes in LIST format defines and installs your new set. It uses program lines 0 and 30000-31300; therefore, your program can't contain these line numbers. It also uses the variable names QQ$ to contain the string data and BASE to hold the location of the new character set in memory; thus, make sure your program doesn't use these names. You can merge the subroutine, in LIST format, with a program already in memory by using the ENTER command.

When you select this option, a prompt tells you to press the RETURN key when you're ready. Diskette users now insert the diskette to be used; cassette users position the tape and press the PLAY and RECORD buttons on their program recorder. All users then press the RETURN key on their computer keyboard.

The program then writes the subroutine to diskette or cassette. It might take a minute or so to write the routine (consider how long it would take you to type in all that data!). The message WRITE DONE displays after a successful operation.

The ORIGINAL prompt redisplays after an UNSUCCESSFUL attempt. In the latter case, check your cassette or diskette to make sure everything is set up correctly and try again.

To cancel the option, press the ESC key instead of the RETURN key after the initial prompt.

**TO MERGE WITH A PREVIOUSLY SAVED PROGRAM**

First, load the saved program into computer memory. Next, cassette users type ENTER "C:", position the tape containing the LISTed subroutine, press the PLAY button on the program recorder and the RETURN key on the computer keyboard.

Diskette users type ENTER "D:LOADSET.LST" (which is the filespec the WRITE subroutine assigns to the LIST file), and press the RETURN key. The character set subroutine then merges with the program in memory.

One final step is absolutely necessary. To order the variable table, which has to be done to make the combined program work, you must then:

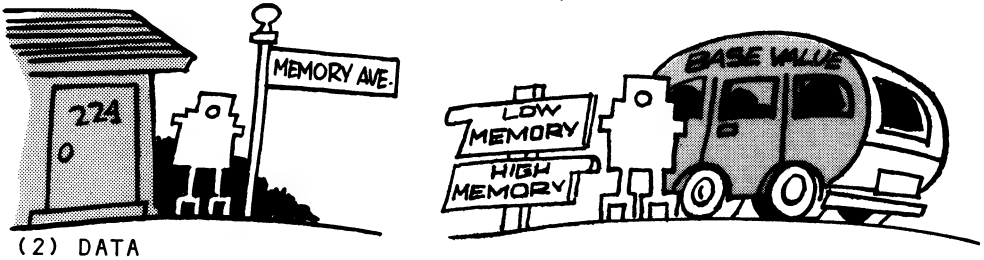
(1) LIST the combined program to diskette or cassette by typing LIST "C:" or LIST "D:filespec", as appropriate.

(2) Type NEW after the LIST is complete.

(3) Enter the program into computer memory by typing ENTER"C:" or ENTER "D:filespec", as appropriate.

You're now ready to SAVE or CSAVE and RUN your combined program.

A final word of caution. The pointer at memory location 756 (decimal), which tells the computer where the character set is stored, is reset to the default value of 224 after every GRAPHICS command. To retain the use of your new character set, this pointer must contain the location of your set. As stated earlier, that location is stored in the variable BASE. Therefore, after every GRAPHICS command, insert the statement POKE 756,BASE to keep your new set in use. You may switch back to the ATARI computer's character set at any time in your program by inserting the statement POKE 756,224.



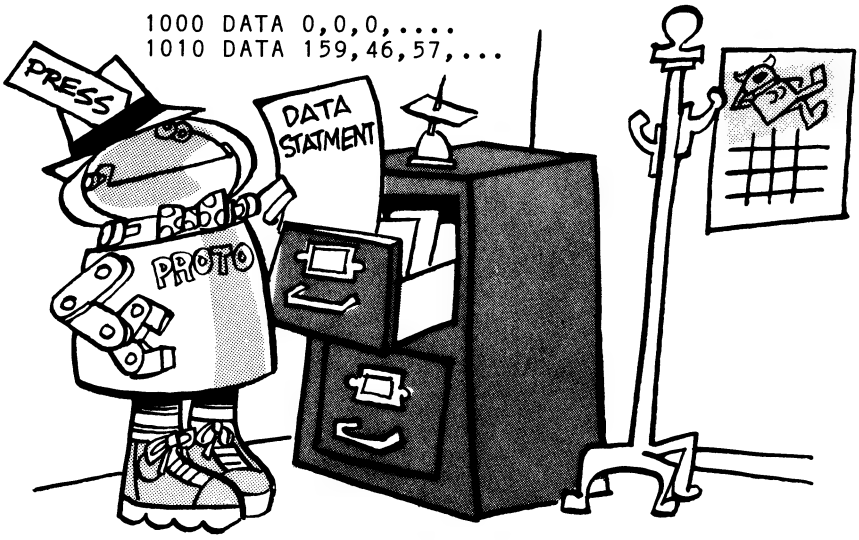

(2) DATA

Use this option to save any character or group of characters to diskette or cassette in the form of numbered BASIC DATA statements. This option is useful in cases where you have only a couple of characters to be altered and don't want to use the memory-intensive subroutine produced by the BASIC option. It's also suitable for occasions when you want to distribute printed listings of your program. Because the BASIC option stores the character data as ATASCII characters rather than as numbers, it's very difficult to produce readable listings of programs using that subroutine. The DATA option is also ideal for using the characters created with INSTEDIT for player missile graphics. Because both players and text characters are eight bits wide, character data can be POKEd into player memory without alteration, and it will produce a player the same shape as the character.

After selecting the DATA option, a prompt displays for entering the name of your data file, if a disk drive is connected, under which the program will save your character set data. The program automatically adds the extender .DAT to your file name so that you'll recognize it as a DATA statement file. Type in your desired file name and press the RETURN key.

Next, the program prompts all users to enter the starting line number for the statements and then the increment between line numbers. If, for example, you enter 1000 as your starting line number and 10 as your increment value, the resulting file will contain statements in the form:

```
1000 DATA 0,0,0,....
1010 DATA 159,46,57,...
```



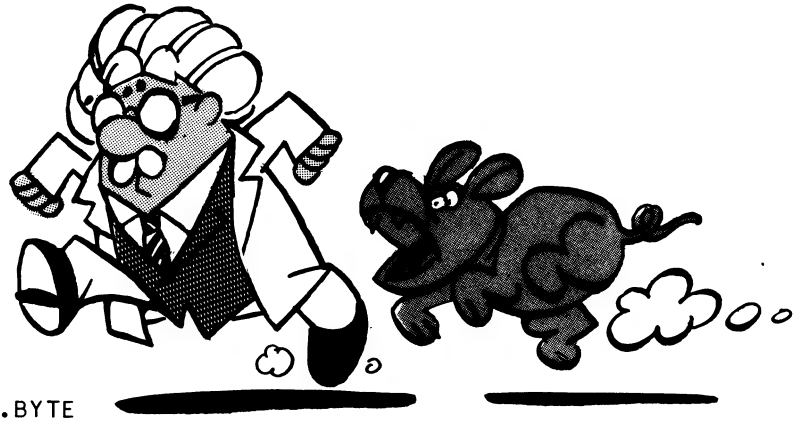

The program then asks users to press the key of the first letter to be saved to the file and the number of consecutive letters appearing directly after that letter that are to be written to the file. Consecutive letters mean the order of the Internal Character Set (corresponding to the order of the complete character set in the middle area of the INSTEDIT display), not ATASCII order. A listing of consecutive order is on page 55 of your ATARI BASIC Reference Manual. Hence, if you want to write a file with data for the letters A through Z, you first press "A" and then "26" to specify the number of characters to be saved. To write a file containing the whole set, you first press the SPACE BAR (which is the first character in the Internal Character Set), and then type 128 for the number of characters.

After you enter this information, the prompt HIT RETURN WHEN READY displays. You can now insert the proper diskette or cassette. When you press the RETURN key, the program will write the selected characters to the file in LIST format. The message WRITE DONE displays after the program completes the write activity.

The original prompt, HIT RETURN WHEN READY, redisplays if the program can't complete the activity.

You can cancel the option by pressing the ESC key instead of the RETURN key.

The program stores character data in groups of 24 bytes (three characters) per program line. You can merge the completed data file with another BASIC file by loading the latter and then using the ENTER command.

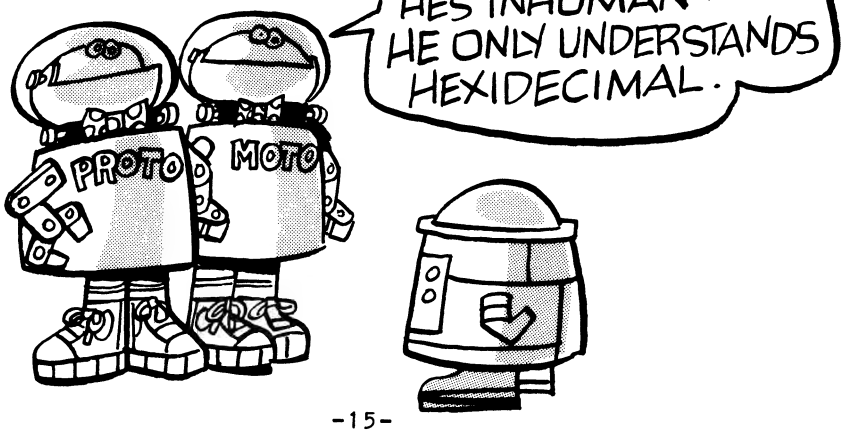


(3) .BYTE

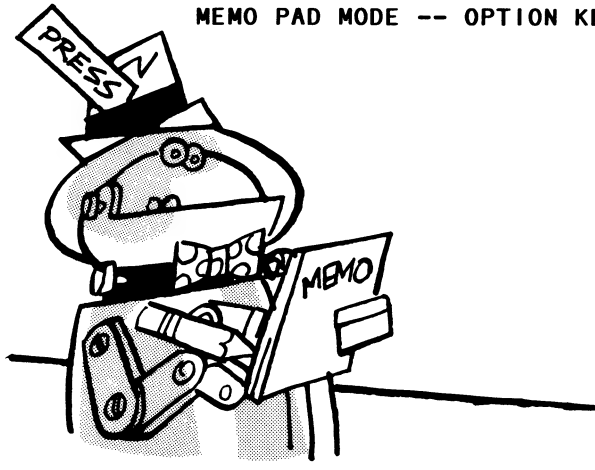
Use this option to produce data files suitable for immediate integration into assembly language source code. The option is similar to the DATA option. Instead of writing the data file in the form of numbered BASIC DATA statements, this option, however, produces numbered lines beginning with the .BYTE directive and followed by the data in hexidecimal form. The output produced takes this form:

1000 .BYTE $00, $49, $2F, $FC, $4B, $0B, $BE, $C7

You can enter the data file directly to the Editor using the Command ENTER #C: or ENTER #D:filespec,as appropriate. Diskette files produced by this option have the extender .BYT . The .BYTE option writes only one character's (eight bytes) worth of data per line.



## MEMO PAD MODE -- OPTION KEY




Press the OPTION key to enter and exit Memo Pad mode. This mode, like the ATARI memo pad, lets you print any combination of characters just to see how they look. INSTEDIT, however, doesn't limit you to graphics mode 0. You can choose any of the six character modes in which to print combinations of characters.

After pressing the OPTION key, a prompt tells you to type a number between 0 and 7. Type in 0, 1, or 2 to set up the bottom half of the screen in the BASIC graphics mode of the corresponding number. Type in 3, 4, or 5 to set the Memo Pad to the corresponding IR mode. Type 6 to use the Memo Pad but not clear the bottom of the screen, which usually occurs when you choose a new mode. This last choice lets you go back and forth from EDIT mode to Memo Pad without changing the contents of the Memo Pad. Type a 7 to restore the lower screen to its initial configuration.

After you choose a mode for the memo pad, the program sets the lower half of the screen to that mode. The cursor, which is visible only in BASIC mode 0 and IR mode 3, is positioned in the upper right-hand corner of the pad. You can then print characters and set up various combinations of edited characters to see their interconnection. You can use a group of characters to form one large picture, or you can use several characters stacked vertically to create a tall player.

To return to the EDIT mode, press the OPTION key again. The lower half of the screen remains the same while you edit characters, until you press the OPTION key again. Once you press OPTION, you clear the lower screen by choosing a mode.

By pressing "7" you'll go back to the screen shown at the start of INSTANT EDIT, but your characters won't be destroyed.

## APPLICATION NOTES

(1) INSTEDIT allows you to view edited characters as they appear in IR modes 3,4, and 5. Although a complete treatment of the Display List is beyond the scope of this manual, a couple of short BASIC routines are included below which will set up a full screen of each of the IR modes, and to use the Script set included with INSTEDIT, which requires the use of IR mode 3.

In each case, start with:

```
10 GRAPHICS 0: DL=PEEK(560)+256*PEEK 561
```

**For IR Mode 3:**

```
20 POKE DL+3,67:FOR I=6 TO 23: POKE DL+1,3: NEXT I: POKE DL+24, PEEK(560): POKE DL+25, PEEK(561): POKE DL+26, 65
```

**For IR Mode 4:**

```
20 POKE DL+3, 68: FOR I=6 TO 28: POKE DL+I, 4:NEXT I
```

**For IR Mode 5:**

```
20 POKE DL+3, 69: FOR I=6 TO 16: POKE DL+I, 5: NEXT I: POKE DL+17, PEEK(560): POKE DL+18, PEEK(561): POKE DL+19, 65
```

(2) Although the Option 1 Write routine is very good for installing your set in a BASIC program, it takes up a fair amount of memory. For those disk users who need to save memory, or want to switch several sets during a single program, I would suggest that you use the initialization routine provided by the LOADSET.LST file, but substitute the Load routine of INSTEDIT (lines 2570-2610) for the definition lines (30100-31200). Remember to change the filespec in the OPEN statement to the name of your set data file.

(3) One way of getting more colors on a Mode 0 or Mode 8 screen is by using color artifacts. You will notice that if you put dots only in odd or even columns (i.e., without ever putting two dots side-by-side), you will generate a couple of new colors. While it is recommended that you avoid this for text characters, it might be useful for graphics characters.

(4) One promising application that seems to be a natural would be to design a series of characters depicting the face cards of a deck of playing cards.

(5) Using the data generated by Write Option 2, it is fairly easy to plot out Mode 0 characters on a Graphics Mode 8 screen, so that text and map-plot graphics can appear on the same horizontal line. As an example, use Option 2 to generate a file with DATA statements for 26 letters. Type NEW, ENTER the data, then type in the following program:

```
10 GRAPHICS 8: DM=PEEK(560) +256 * (PEEK(561)+4:
DM=PEEK(DM) + 256*PEEK (DM+1): OFF=1680

20 FOR I=1 TO 25: FOR J=0 TO 7: READ A: POKE DM +OFF
+J*40 +I,A: NEXT J: NEXT I

30 GOTO 30
```

By varying the offset constant (OFF), you can position these characters around the screen.

(6) As mentioned in the Menu section, the data generated by Write Option 2 or 3 can be used directly to define Players for Player-Missile Graphics. This data can be POKEd directly into the area reserved for Player data. Just remember that data from more than one character will stack up vertically, one byte per horizontal line. Also, to save space and time, remove any 0 data that precedes or follows the actual player shape.

You may want to keep in mind that Player-Missile Graphics offers three width options for each Player. Therefore, you may make use of INSTEDIT'S display of the double-wide characters of Graphics Modes 1 and 2, as well as the large center display (which itself is 1 maximum-width player) to visualize the appearance of the Players you design in varying widths.

## TROUBLESHOOTING

(1) Normally, about the only type of problems encountered will be related to Input/Output. Examples of such errors are use of illegal disk file names, trying to write to locked files, trying to read non-existent files , or trying to read an improperly-positioned tape. Instedit does not return an error message when a load or save fails, but rather returns to the last prompt. This gives you an opportunity to check your cassette or disk, make sure everything is set up right, and try again, or cancel.

(2) At most times the user is protected against accidental use of the editing keys in a way which would interfere with the screen display. However, full editing functions are restored when the user is requested to input information longer than one character, such as a file name. During these times, accidental use of an editing

function such as the line delete may disrupt the display. If this occurs, complete the function in process. Then, hit the OPTION key to enter Memo Pad mode. Select option 7, Exit. This will restore the screen to its original configuration.

(3) While the Break key has been disabled, the System Reset key still works. If you accidentally hit Reset, you can restart the program by typing RUN. If you do, however, all previous character data will be lost. In order to save your data, before typing RUN type GOTO 2000. This will let you enter the Save routine without clearing character data. When the "SAVE COMPLETE" message appears, you may hit Reset and RUN the program.

## ABOUT THE PROGRAM

In order to understand how INSTEDIT works, a fair understanding of some of the special hardware features of the ATARI computers is necessary. In particular, INSTEDIT makes use of Player-Missile Graphics, mixed Display Lists and character-set indirection . A review of some of the literature dealing with these topics will go a long way toward aiding an understanding of how the program works. References to some of these sources appear in the Selected Bibliography at the end of this manual.

While not a line-by-line analysis of the program, the following general description should give you an idea of the program's logic and organization.

The initialization routine starts at the back of the program, at line 30000. Variables are set up for constants, the keyboard is opened as a device, and the Handler Table is checked to see if a disk device is present (30000-30030). Next, strings are dimensioned. The most important of these are C$, which will hold the new character set data, and PM$, P0, P1$, etc., that hold the Player Missile Graphics data. Since both P-M information and character sets must start on a 1k boundary, FILL$ is first dimensioned to a length that is sufficient to waste the space between the last string data (ADR(D$) and the next highest 1K boundary (30060).

By superimposing the P-M graphics data area and the character set data area on the memory area reserved for these strings, we can take advantage of the machine-language routines in BASIC that move string data around so quickly. In addition, it enables the use of the XIO 7 and 11 commands, which move blocks of 255 bytes. These commands are tied to the use of strings, and they account for the ability of the program to rapidly transfer data using only BASIC routines.

After the strings are initialized, the ROM character set is copied to the area of RAM set aside for C$ (30100),

using a short machine-language routine. The Break key is disabled, the mixed-mode display is created, and the initial screen graphics are drawn (30100 to 31365). Player-Missile Graphics are then used to create the center frame (Player 2 and Missiles grouped as Player 4), the large display character (Player 1), the cursor (Player 0), and the grid background (Player 3). The priority register is set so that when the cursor and large characters overlap, the cursor changes color . Temporary strings are used to transfer data to Player memory, and P-M graphics are enabled (31370-31395). After initializing a couple of short machine-language routines to shift characters and to draw the large center display, the program proceeds to the main loop at Line 200.

The main loop (200-290) checks the keyboard, the console switches, the joystick, and the trigger button, each in turn. First to be checked are the Write (205), OPTION (207), Edit (210), Save (215), and Load (220). If one of these has been hit, the program jumps to the appropriate subroutine. In the initial phase, if one of these has not been hit, the program branches to the routine that alternates the menu colors (300-340) and then loops to 200.

If in the Edit mode, the arrow keys are checked, and if one is hit, the Shift routine contained on the same line is executed (235-250). Atari (259), Blank (255), Invert (260), Mirror (254), Restore (270) and Twist (252) are also one-line routines, while Copy (265) branches to a two-line subroutine at 450, very similar to the Edit subroutine at 400.

Next, the stick is checked, and if not used the program loops back to 200. If the stick has been used, the cursor is moved by using the horizontal position registers, or moving the string information up as appropriate.

The trigger check occurs at line 230. If the button has been pushed, the program branches to the subroutine that actually sets and erases the dots (350-390). This routine checks the collision register to see if Players 0 (the cursor) and 1 (the large character) overlap. Next, the bit position and byte number of the character are calculated based on the horizontal and vertical position of the cursor. Finally, 2 is raised to the power represented by the bit place, and either added or subtracted from the byte total, depending on the value of the collision register.

At this point, we should mention the subroutine at Line 10. This is the routine that draws the large character, and because it is so frequently used, it is the only subroutine which has been placed in front of the main loop.

The subroutine at 800-880 sets up the display of the model characters, by POKEing them into display data. The Memo Pad routine is at 925 to 995. Save and Load routines are at 2000 and 2500 respectively, and the Write routines finish the subroutine section at 3000-3910.

## SELECTED BIBLIOGRAPHY

The following magazine articles will aid the reader in understanding the subjects of character set editing and Player-Missile Graphics:

"Outpost:Atari--Missile Graphics Mysteries Revealed", George Blank, Creative Computing, January 1981, p.176

" Character Generation on the Atari", Charles Brannon, COMPUTE!, February 1981, p.76

" Player-Missile Graphics with the Atari PCS", Crawford, COMPUTE!, January 1981, pp. 66-77

" An Introduction to Atari Graphics", Crawford and Winner, BYTE, January 1981, pp.18-32

" Designing Your Own Atari Character Sets", Gary Patchett, COMPUTE!, March 1981, p.72

Also recommended:

"MASTER MEMORY MAP", 100's of important memory locations for the ATARI 400/800 by Educational Software, © 1982

"ATARI PERSONAL COMPUTER SYSTEM HARDWARE MANUAL" available from ATARI Customer Service. This manual gives complete details concerning hardware features. Appendices A and B, which deal with Player-Missile Graphics and mixing Graphics Modes, will be of particular interest.


In addition to the articles above, the following software packages are recommended for a more in-depth study of the subject material.

"TRICKY TUTORIAL # 5: Player Missile Graphics", Learn to write your own animated games from Educational Software, 1982

"TRICKY TUTORIAL # 8: Character Graphics", contains the best character graphics editor for the ATARI 400/800 as well as valuable instruction. By Educational Software, 1982

## APPENDIX

## ATARI'S "HIDDEN" CHARACTER MODES

The reason that Atari computers are able to offer such a large number of Graphics modes is that they use a separate graphics microprocessor to handle the screen display. This chip receives instructions on how to display data from a simple program contained in RAM. If you are using BASIC, the computer writes this program every time you give a GRAPHICS command. But since this program is in RAM, it is possible for the user to alter it with the POKE command. Several articles have appeared in magazines giving instructions on how to change this program, called the Display List, so that many different graphics modes can appear on the screen at the same time. A step-by-step plan for creating these mixed Display Lists appears in Appendix B of the Atari Hardware Manual (available from Atari Customer Service) . The key step involves changing the DISPLAY instruction. This instruction consists of a number from 2 to 15. Each time it appears, it orders the graphics chip to display one line of a particular mode. For example, POKEing in a 2 orders the chip to display one line of Mode 0, a 6 orders up one line of Mode 1, a 7 one line of Mode 2, etc. Notice how the numbers 3,4,5, were skipped? These are the DISPLAY instructions for the "hidden" character modes.

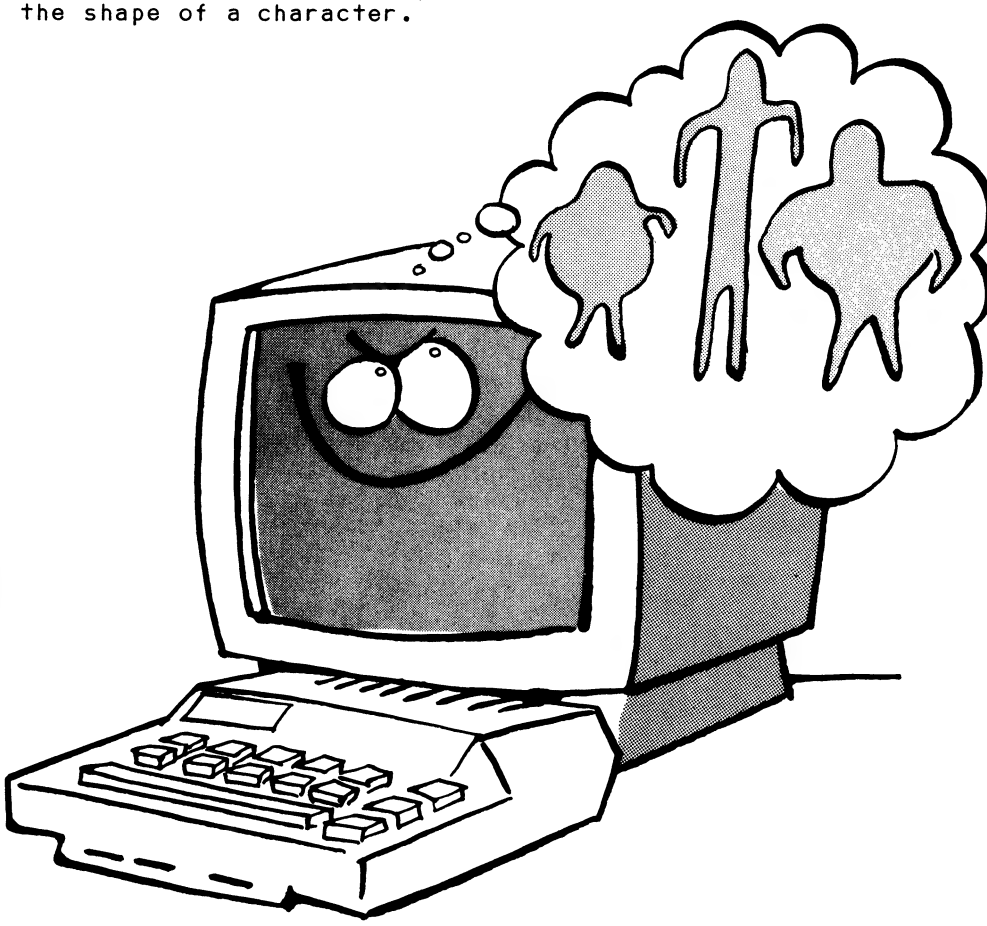
When I first tried POKEing those DISPLAY instructions into a Mode 0 Display List, I came up with some pretty funny-looking characters that I really didn't understand. In order to find out the purpose behind these modes, which are not supported by BASIC, I had to turn to the Hardware Manual. It outlines, in fairly technical terms, some hardware features which are not explained by the reference material supplied with the computer.

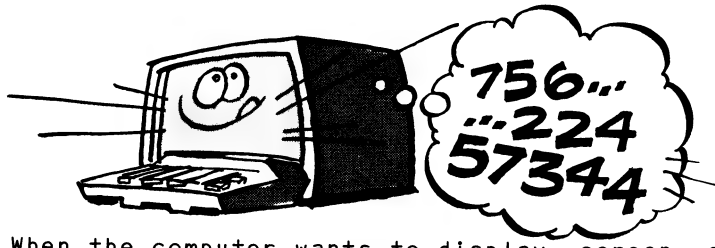
To help in the explanation of these modes, I have listed two short programs. The first of these (Listing 1) demonstrates what is referred to in the Hardware Manual as Instruction Register (IR) Mode 3. In Line 10, I POKE a 3 into bytes 19-26 of the Display List, producing a screen which is 1/2 BASIC Graphics Mode 0 and 1/2 IR Mode 3. Next, the whole character set is printed in both modes (Line 30). Finally, a few adjacent characters are printed in both modes for the purpose of comparison (Lines 40-45).

When this program is run, the IR Mode 3 characters at the bottom of the screen do not appear to be much different than the mode 0 characters at the top. On more

careful examination, however, some differences can be detected. First, there is more room between the rows of characters in IR Mode 3. This is seen in the fact that while the four diagonal graphics characters in the middle of the screen form a diamond shape in Mode 0, there is a gap between the top and bottom triangles in IR Mode 3. Also, the cursor is taller in that mode. The second difference occurs only in the last 32 characters of the IR Mode 3 set. These characters appear to be shifted, so that the top part of the character has been cut off and moved below the bottom of the character.

According to the Hardware Manual, there is a simple reason for these differences. By creating a longer block for these character, and having some appear at the top of the block, and some at the bottom, the user can create a custom character set with true descenders for lower case letters like "y" and "p". In order to explain exactly how this mode accommodates these changes, however, we must first review the method by which the computer determines the shape of a character.






When the computer wants to display screen data as a graphics character, it must look up the shape of the character in a table stored in memory. To find this table, it looks up its location in a pointer held in memory location 756 (decimal). Normally, this pointer holds the number 224, because the ROM character data is stored starting at page 224 (this is equal to memory location 57344, decimal). If you are using a custom character set, such as those created with INSTEDIT, the pointer will hold the beginning location of that set of data in RAM. Each character is represented by 8 bytes of data. As each of these bytes is composed of 8 binary digits (or bits), we can picture this data in the form of an 8x8 grid. Figure 1 shows how the data for the upper and lower case letter "L" is interpreted into the character seen on screen. In this drawing, each horizontal row represents one byte (the numeric value of which is given on the left). Each vertical column represents a bit place. A darkened square represents a "1" state in the corresponding bit location (the bit values, which equal the successive powers of 2 from 2 to the 0 (1) to 2 to the 7th (128) are shown at the top of each column). So, for example, no squares are darkened in the top row of Figure 1(a), and therefore the first byte has a value of 0. In the second through sixth rows, where bits 5 and 6 are darkened, the byte value is 96 (64+32).

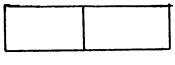
In IR Mode 3, however, these same characters are set up in a 10x8 grid. Two blank scan lines are inserted below each of the first 96 characters (see Figure 2(a)). The last 32 characters, which include the lower-case alphabet, receive special handling. When one of these characters is set up in the grid, the first two bytes are shifted down to the bottom two lines (see Figure 2(b)). This shift of the last 32 characters means that they use the bottom 8 lines of the grid, while the other characters use the top 8 lines, thus enabling the use of the bottom two lines for descenders. A practical example of the use of this mode can be seen in the Script set which is included with INSTEDIT. You will note that while the mode accommodates the characters with descenders very well, the tall characters such as the lower-case "b" and "d" must be transferred to the spot reserved for their respective Control-characters, so that they will not be shifted down, and can attain their full height.

The other two hardware-supported character modes are demonstrated by the program in Listing 2. Lines 10-20 of that program set up the screen half in IR Mode 4, and half in IR Mode 5. Line 30 prints the full character set in each mode. Line 40 changes the background color for better visibility. The rest of the program enables the use of the console switches to change the color and luminescence value of each register. The SELECT button determines the register, START changes the color of that register, and OPTION the brightness.

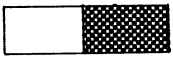
These two modes are 4-color character modes. The only difference between these two modes is that IR Mode 5 characters are twice as high as those of IR Mode 4. Unlike BASIC modes 1 and 2, these IR Modes can mix colors within a particular character.

Upon looking at the standard character set in these two modes, it will be apparent that these modes are not really suitable for text characters. When used with custom-designed graphics character sets, however, four-color graphics with the same resolution as BASIC Graphics mode 7 can be created, and placed on the screen just by PRINTing a string of characters.

To enable this colorful display, the computer divides each byte of character display data into 4 groups of 2 bits each. These groups determine the color of the four pixels per row. The four possible combinations produce the following colors:



Neither bit set-- displays the background color (register 4)



Right bit set--displays the color in register 0

Left bit set--displays the color in register 1

Both set--displays the color in register 2 for normal video, and the color in register 3 for inverse video characters.

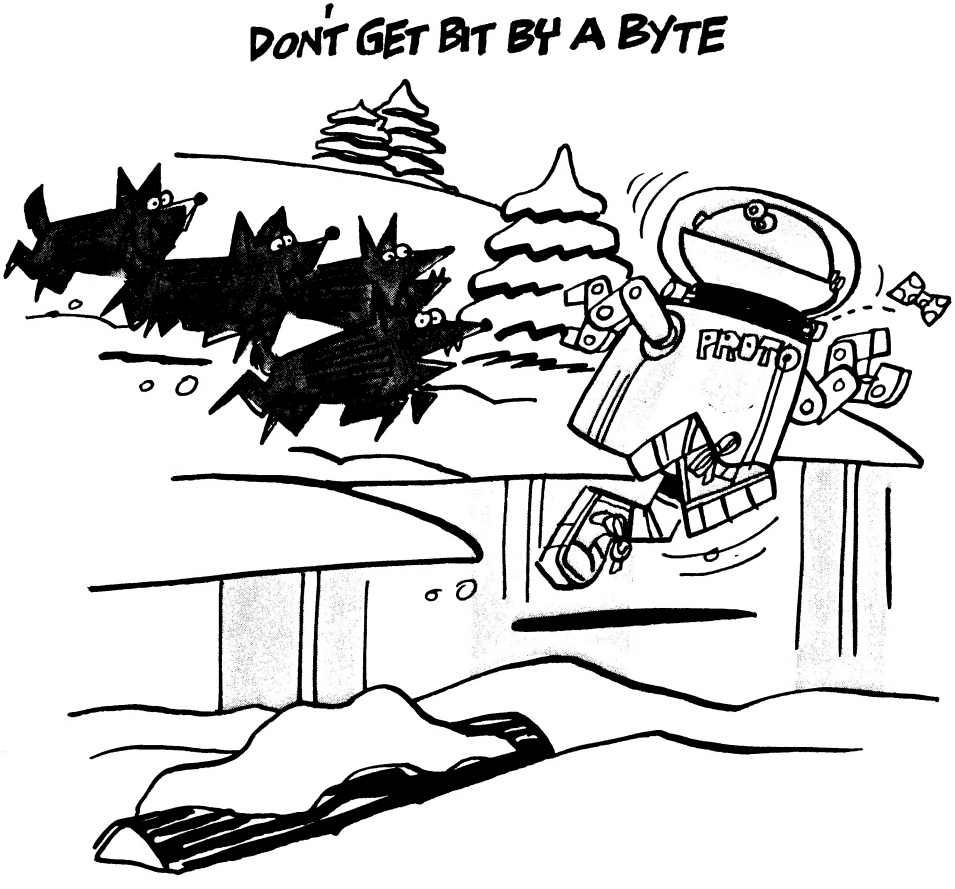
Because two bits are needed to determine the color of each pixel, the horizontal resolution is cut in half. Figure 3 shows how this affects letters in the existing

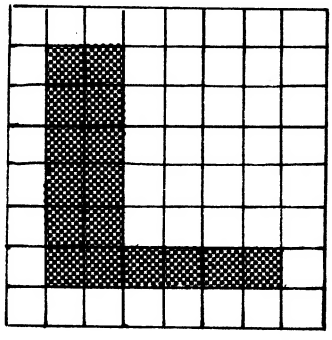
character set. You should be able to verify this effect by changing the color registers in the demonstration program by using the console switches as explained above.

```
0 REM ** LISTING 1
1 REM ** DEMO OF IR MODE 3
2 REM ** by Sheldon Leemon
3 REM **
5 REM ** SET UP MIXED-MODE SCREEN
6 REM **
10 ? CHR$(125):X=PEEK(560)+PEEK(561)*256+19:FOR I=0 TO
7:POKE X+I,3
20 NEXT I:POKE X+8,65:POKE X+9,PEEK(560):POKE
X+10,PEEK(561)
21 REM *
25 REM * SET UP COMPARISON CHARACTERS
26 REM *
30 GOSUB 60:POSITION 2,17:GOSUB 60
40 POSITION 10,12:? CHR$(6);CHR$(7)
41 POSITION 10,13:? CHR$(7);CHR$(6);"  LI";CHR$(160)
45 POSITION 10,14:? CHR$(6);CHR$(7);"     ";CHR$(160);"LI"
46 POSITION 10,15:? CHR$(7);CHR$(6):POSITION 15,10:? " "
50 POKE 752,1:POSITION 2,9:? CHR$(28)
51 REM *
55 GOTO 55
56 REM *
60 FOR I=0 TO 127:? CHR$(27);CHR$(I);:NEXT I:RETURN
```

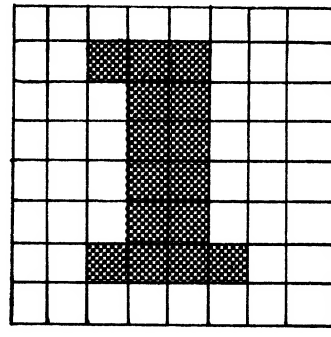
```
0 REM ** LISTING 2
1 REM ** DEMO IF IR MODES 4 AND 5
2 REM ** by Sheldon Leemon
3 REM **
5 REM ** SET UP MIXED-MODE SCREEN
6 REM **
10 ? CHR$(125):X=PEEK(560)+PEEK(561)*256+3:POKE X,69
15 FOR I=3 TO 8:POKE X+I,5:NEXT I:FOR I=9 TO 16:POKE
X+I,4:NEXT I
20 POKE X+19,65:POKE X+20,PEEK(560):POKE
X+21,PEEK(561):POKE 752,1:? ""
21 REM *
25 REM * PRINT CHARACTER SETS
26 REM *
30 GOSUB 60:? :? :GOSUB 60:POSITION 0,0:?
CHR$(156):POSITION 1,13
31 REM *
35 REM * CHANGE BACKGROUND COLOR
36 REM *
40 FOR DELAY=1 TO 1500:NEXT DELAY:? CHR$(253):SETCOLOR
4,0,14
```

```
41 REM *
45 REM * COLOR REGISTER CHANGE ROUTINE
46 REM *
50 R=0:S=5:GOSUB 70
52 S=PEEK(53279):IF S=5 THEN R=R+1-5*(R=4):GOSUB 70
54 IF S=6 THEN C=C+1-16*(C=15):SETCOLOR R,C,L:GOSUB 75
56 IF S=3 THEN L=L+2-16*(L=14):SETCOLOR R,C,L:GOSUB 80
58 FOR DELAY=1 TO 50:NEXT DELAY:GOTO 52
60 FOR I=1 TO 154:? CHR$(27);CHR$(I);:NEXT I
65 FOR I=156 TO 255:? CHR$(27);CHR$(I);:NEXT I:RETURN
70 M=PEEK(708+R):C=INT(M/16):L=M-16*C
71 POSITION 2,15:? "REGISTER ";R:GOSUB 75:GOSUB 80:RETURN
75 POSITION 15,15:? "COLOR ";C;"  ":RETURN
80 POSITION 25,15:? "LUM. ";L;"  ":RETURN
```



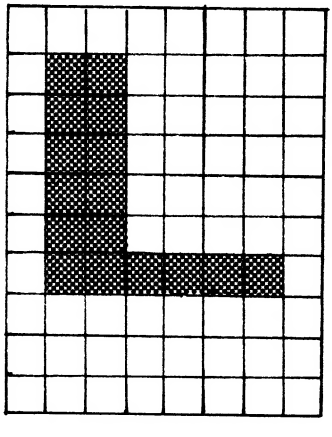


(a)

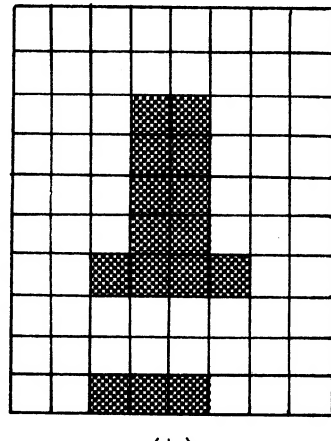


(b)

FIGURE 1

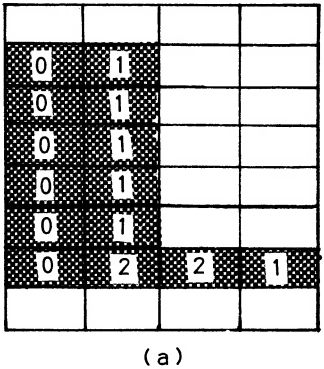


(a)

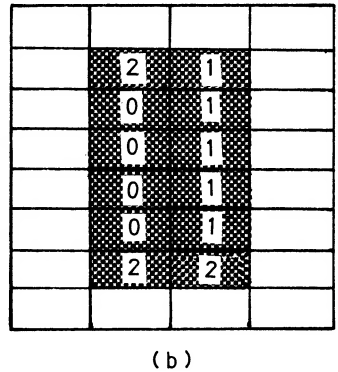


(b)

FIGURE 2



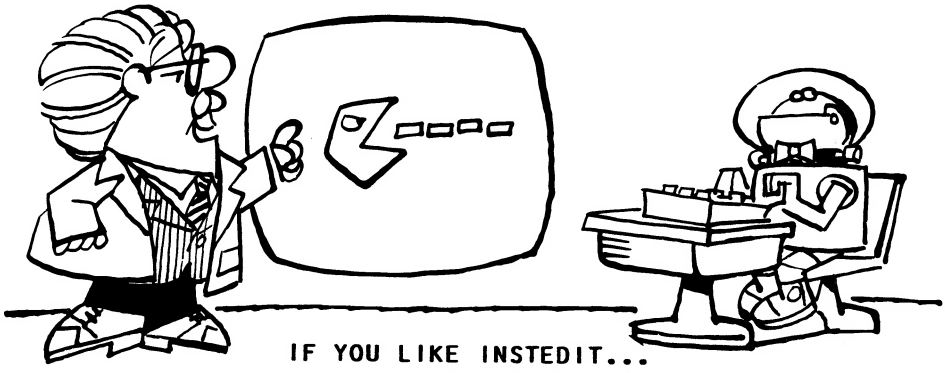

(a)




(b)

FIGURE 3

**The numbers in the shaded figures show the color register displayed.**



**IF YOU LIKE INSTEDIT...**

You'll love Character Graphics, Tricky Tutorial(tm) #8. With Professor Von Chip's help, it's an easy step from editing the characters in your ATARI to using them in your own programs, even arcade-style games.

There are Tricky Tutorials to help you with everything from how to use your computer to programming your own special sound and graphic effects. The Professor and Proto uncomplicate these functions and make learning to program fun!

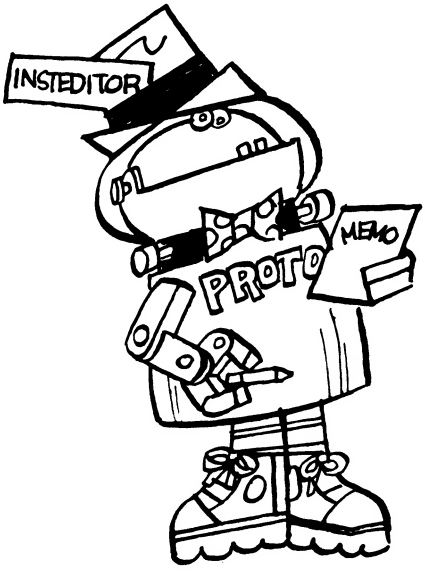
#1 - DISPLAY LISTS (Mixing many Graphics Modes on the screen at once.).
#2 - SCROLLING (Move Text and Graphics up and down or across your screen smoothly.)
#3 - PAGE FLIPPING (Program screen changes to switch instantly, like flipping pages in a book.)
#4 - BASICS OF ANIMATION (A beginner's guide to creating animated shapes and simple games.)
#5 - PLAYER MISSILE GRAPHICS (Learn to write arcade-style games.)
#6 - SOUND & MUSIC (From single notes to songs and special effects, all explained clearly.)
#7 - DISK UTILITIES (Seven utilities to help you use your disk drive all wrapped up in this one package.)
#8 - CHARACTER GRAPHICS (See above.)
#9 - GTIA- GRAPHICS 9 to 11 (A veritable rainbow of nine colors and sixteen shades makes your screen come alive!)
#10 - SOUND EFFECTS (Tweets, whistle, and roars from your Atari? Yes, it's all possible with this Tutorial.)
#11 - THE MEMORY MAP TUTORIAL (Professor Von Chip's explains his favorite tool to help you utilize your Atari's memory locations.)

...And look for new Tricky Tutorials coming soon to your favorite software store!

# INSTEDIT

A complete character editor with an easy-to-understand manual that makes programming your own custom characters easy and fun!

-- "Instedit is the best set editor we've seen"
-- ATARI APX




**REQUIRES:**

**16K TAPE/32K DISK**
**(and a little imagination)**

PROGRAM EXCHANGE
EDUCATIONAL SOFTWARE inc.

4565 Cherryvale Ave
Soquel, Ca. 95073
(408) 476-4901